第八章　输入输出程序设计

8.1　I/O设备的数据传送方式

8.2　I/O程序举例

8.3　中断传送方式

8.4　80386输入输出

8.5　80386的中断处理

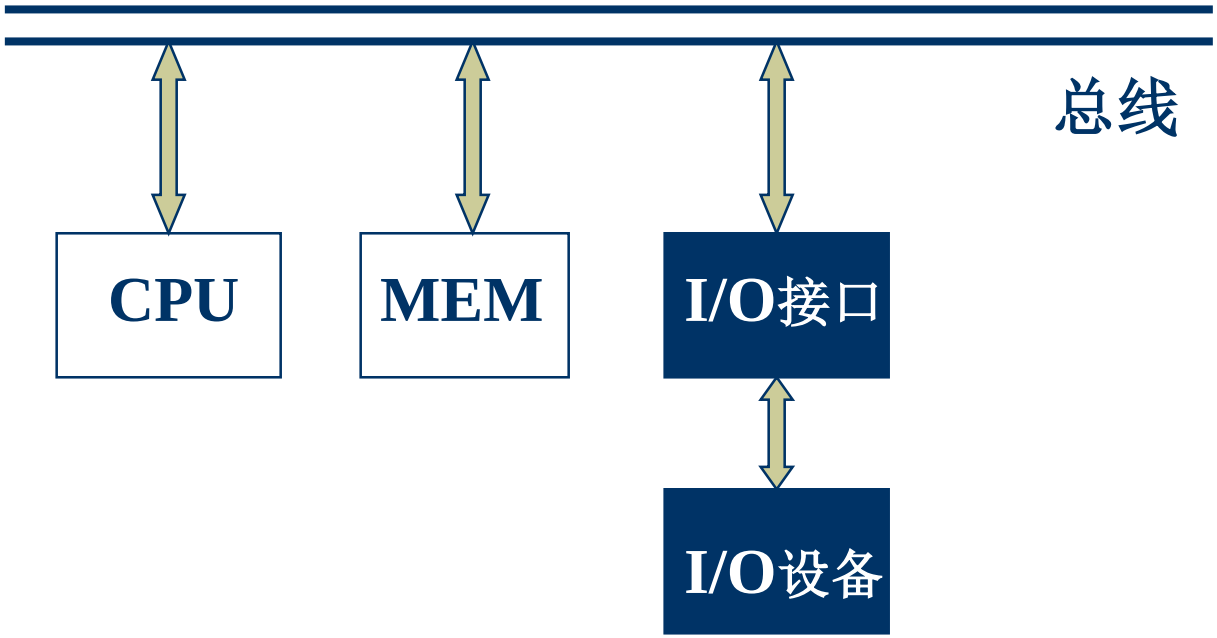
本 章 目 标

1. 了解输入输出程序设计的基本概念

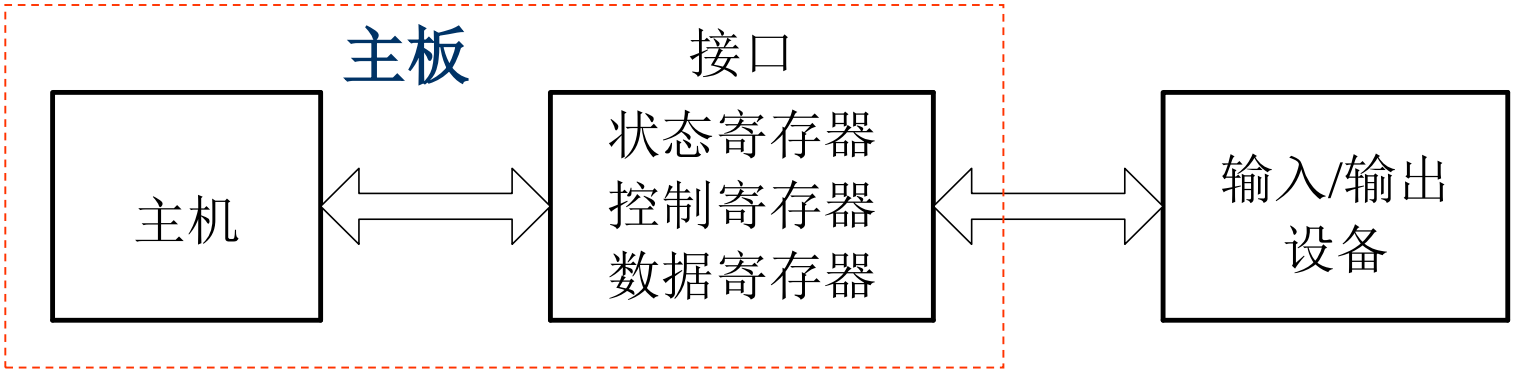
2. 掌握IN/OUT指令的用法

3. 掌握中断传送方式工作机制与编程方法

8.1 I/O设备的数据传送方式



8.1.1 主机、外设与接口



- 每种输入输出设备都要通过一个硬件接口或控制器和CPU相连
  - 如，打印机接口，显示器接口等
- 从程序设计的角度看，接口由一组寄存器组成，是完成输入输出的桥梁

8.1.2 I/O端口
(IO接口上的寄存器)

- I/O端口地址：为了访问接口上的寄存器，系统给这些寄存器分配专门的存取访问地址，这样的地址称为I/O端口地址
- 8086/8088CPU系统中，I/O端口地址和存储单元的地址是各自独立的，分占两个不同的地址空间
- 8086/8088CPU提供的I/O端口地址空间达64KB
  - 可接64K个8位端口(字节)，或可接32K个16位端口(字)
  - PC及其兼容机实际只使用0~3FFH之间的I/O端口地址
- 存取接口寄存器中的数据是依靠I/O指令完成的

X86机器I/O端口地址分配表

I/O 地址	功　能	I/O 地址	功　能
00～0F	DMA 控制器 8237A	2F8～2FE	2 号串行口(COM2)
20～3F	可编程中断控制器 8259A	320～324	硬盘适配器
40～5F	可编程中断计时器	366～36F	PC 网络
60～63	8255A PPI	372～377	软盘适配器
70～7F	实时钟	378～37A	2 号并行口(LPT1 打印机)
80～8F	DMA 页表地址寄存器	380～38F	SDLC 及 BSC 通讯
93～9F	DMA 控制器	390～393	Cluster 适配器
A0～BF	可编程中断控制器 2	3A0～3AF	BSC 通讯
C0～0E	DMA 接口专用	3B0～3BF	MDA 视频寄存器
F0～FF	协处理器	3BC～3BE	1 号并行口
170～1F7	硬盘控制器	3C0～3CF	EGA/VGA 视频寄存器
200～20F	游戏控制端口	3D0～3D7	CGA 视频寄存器
278～27A	3 号并行口(LPT2 打印机)	3F0～3F7	软盘控制寄存器
2E0～ 2E3	EGA/VGA 使用	3F8～3FE	1 号串行口(COM1)

8.1.3 I/O指令

1. **输入指令：IN 累加器，端口地址**
   - IN AL, PORT ;AL←(PORT)
   - IN AX, PORT ;AL←(PORT), AH←(PORT+1)
   - IN AL, DX ;AL←(DX)
   - IN AX, DX ;AL←(DX), AH←(DX+1)
2. **输出指令：OUT 端口地址，累加器**
   - OUT PORT, AL ;(PORT)←AL
   - OUT PORT, AX ;(PORT)←AL, (PORT+1)←AH
   - OUT DX, AL ;(DX)←AL
   - OUT DX, AX ;(DX)←AL, (DX+1)←AH

➢ **累加器：只能使用累加器**

- AX（16位字操作）； AL（8位字节操作）

➢ **端口地址：只有两种方式给出**

- **立即数：** PORT（00H～FFH）；**寄存器间接：** DX（0000H～FFFFH）

8.1.4 I/O设备的数据传送方式

1. 程序直接控制I/O方式
   1). 无条件传送方式
   2). 查询方式
   - 主要缺点：CPU和I/O设备不能并行工作，CPU资源浪费十分严重
2. 直接存储器存取（DMA）方式
3. 中断方式

- 主要考虑因素：
  - ✓ CPU与IO设备速度匹配问题
  - ✓ 减轻CPU负担
  - ✓ 外设请求服务处理的时机

程序直接控制I/O方式

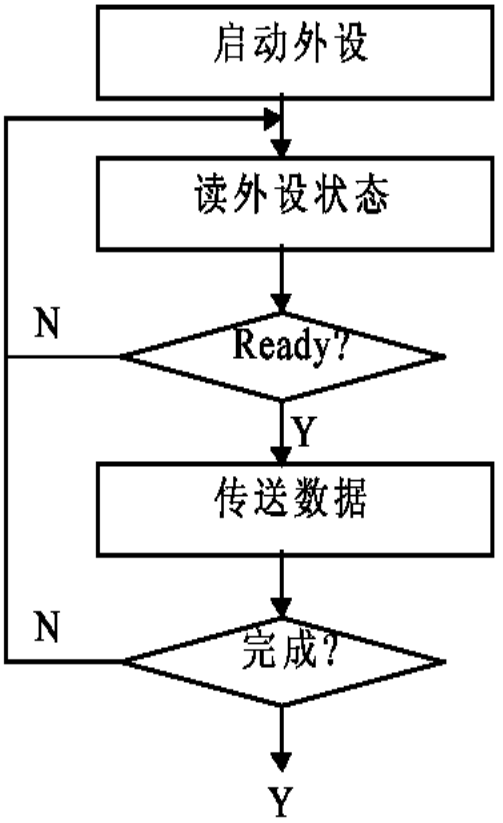
1). 无条件传送方式

- 默认外设总是处在“准备好” 或者“不忙” 状态
- 直接使用IN、OUT指令实现数据传送
- 例：外设输出数据端口地址70H，输入数据端口地址60H，则
  - 字节数据输入：IN AL，60H
  - 字数据输入：IN AX，60H ;(60H)→AL，(61H)→AH
  - 字节数据输出：OUT 70H，AL
  - 字数据输出：OUT 70H，AX
- 无条件传送方式要求（适合情况）
  - 外设的工作速度与CPU同步，否则就会出错
    - 如果CPU与外设的工作速度不同步应采用查询等其它方式
  - CPU负载轻，其它工作少时

程序直接控制I/O方式

2). 查询方式

- 在输入输出之前首先查询外设的状态
  - 当外设状态处于“准备好”（输入方式下）或者“不忙”（输出状态下），才可以进行输入/输出；
  - 反之，要一直等待到外设“准备好”或“不忙”
- 外设接口正在准备数据或者输出任务没有完成之前，其状态处于“没准备好”或者“忙”



- 例如：外设输出数据端口（输出数据寄存器）地址70H，输出状态端口地址72H，输出状态端口最高位=1表示输出设备不忙。则：
  - 查询方式字节数据输出：

```
WAIT: IN   AL, 72H     ; 读取输入口的状态
      TEST AL, 80H     ; 测试状态寄存器的最高位
      JZ   WAIT        ; 若状态位=0，则继续测试等待
      MOV  AL, VAR
      OUT  70H,  AL    ; 输出数据
```

  - 查询方式字输出：

```
WAIT: IN   AL, 72H     ; 读取输出口的状态
      TEST AL, 80H     ; 测试状态寄存器的最高位
      JZ   WAIT        ; 若状态位=0，则继续测试等待
      MOV  AX, VAR
      OUT  70H, AX     ; 输出数据
```

- 例如：外设输入数据端口地址60H，输入状态端口地址62H，输入状态端口62H的最高位=1表示输入设备准备好。则：
  - 查询方式字节数据输入：

```
WAIT:  IN    AL, 62H      ; 读取输出口的状态
       TEST  AL, 80H      ; 测试状态寄存器的最高位
       JZ    WAIT         ; 若状态位=0，则继续测试等待
       IN    AL, 60H      ; 输入数据
         ⋮
```

  - 查询方式字输入：

```
WAIT:  IN    AL, 62H      ; 读取输出口的状态
       TEST  AL, 80H      ; 测试状态寄存器的最高位
       JZ    WAIT         ; 若状态位=0，则继续测试等待
       IN    AX, 60H      ; 输入数据
         ⋮
```

查询方式的问题

- 查询方式问题：虽然通过CPU重复查询外设状态，直到外设准备好再进行数据传送，解决了CPU与外设不同步的问题。可是存在以下问题：
  - 查询等待中的“踏步”问题
  - CPU和I/O处于串行工作方式，CPU效率不高

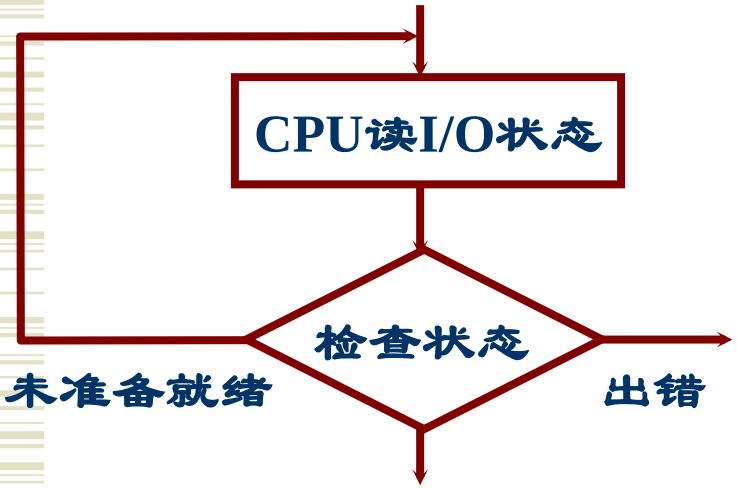
中断方式

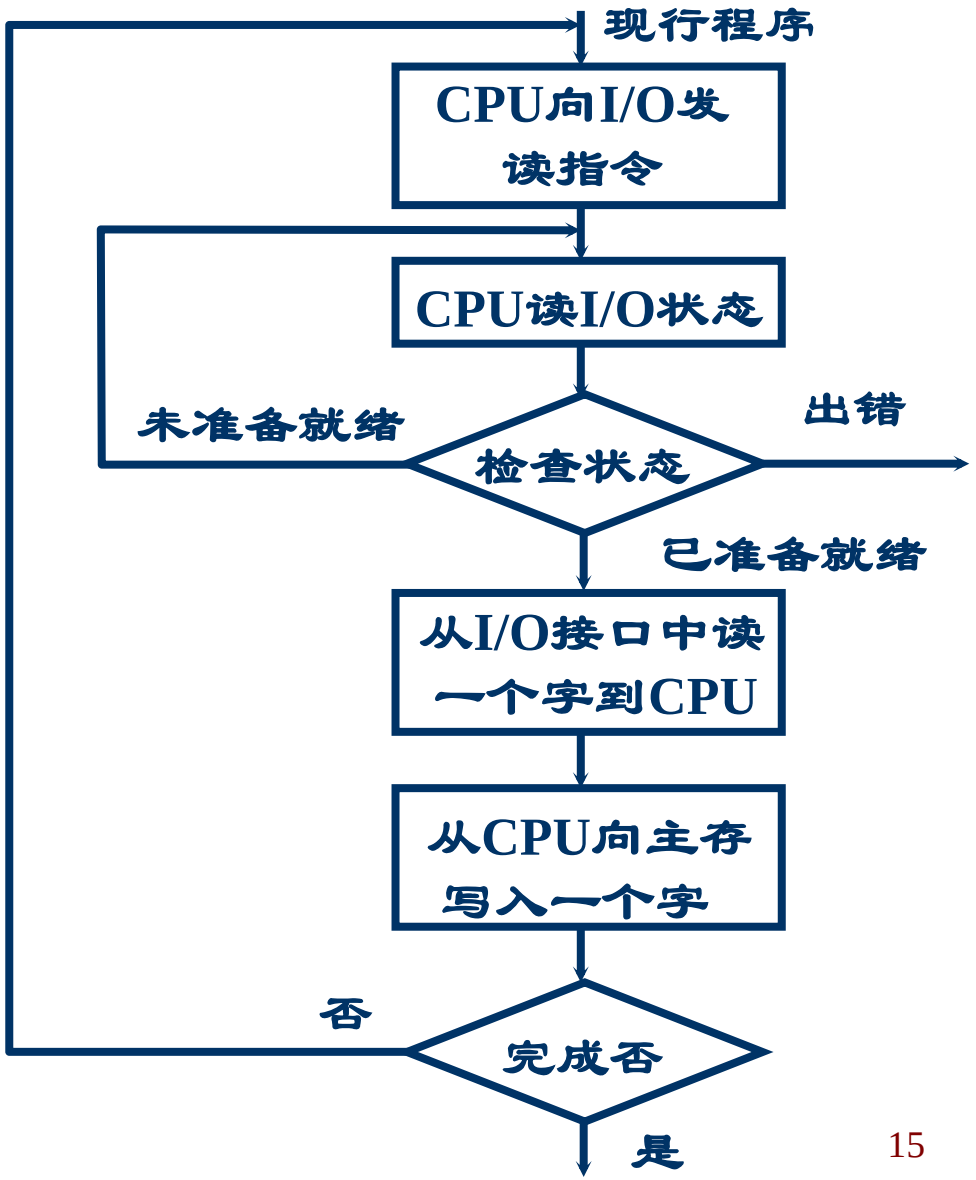
- 中断方式：
  - 当外设准备好时，外设主动向CPU发出中断服务请求，CPU暂时中止现行程序的执行，转入中断服务处理程序完成输入/输出工作，之后返回被中断的程序继续执行
- 中断方式特点：
  - CPU和I/O设备能够并行运行
  - 具有及时处理响应意外事件或异常的能力

I/O 与主机信息传送--程序查询方式

CPU 和 I/O 串行工作

踏步等待

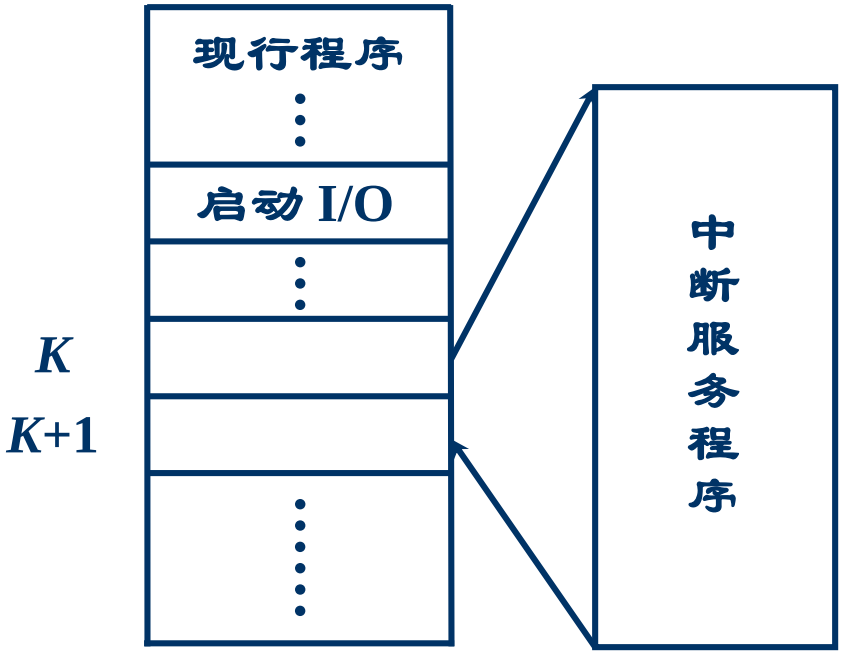




I/O 与主机信息传送--程序中断方式

I/O 工作
- 自身准备　　CPU 不查询
- 与主机交换信息　CPU 暂停现行程序

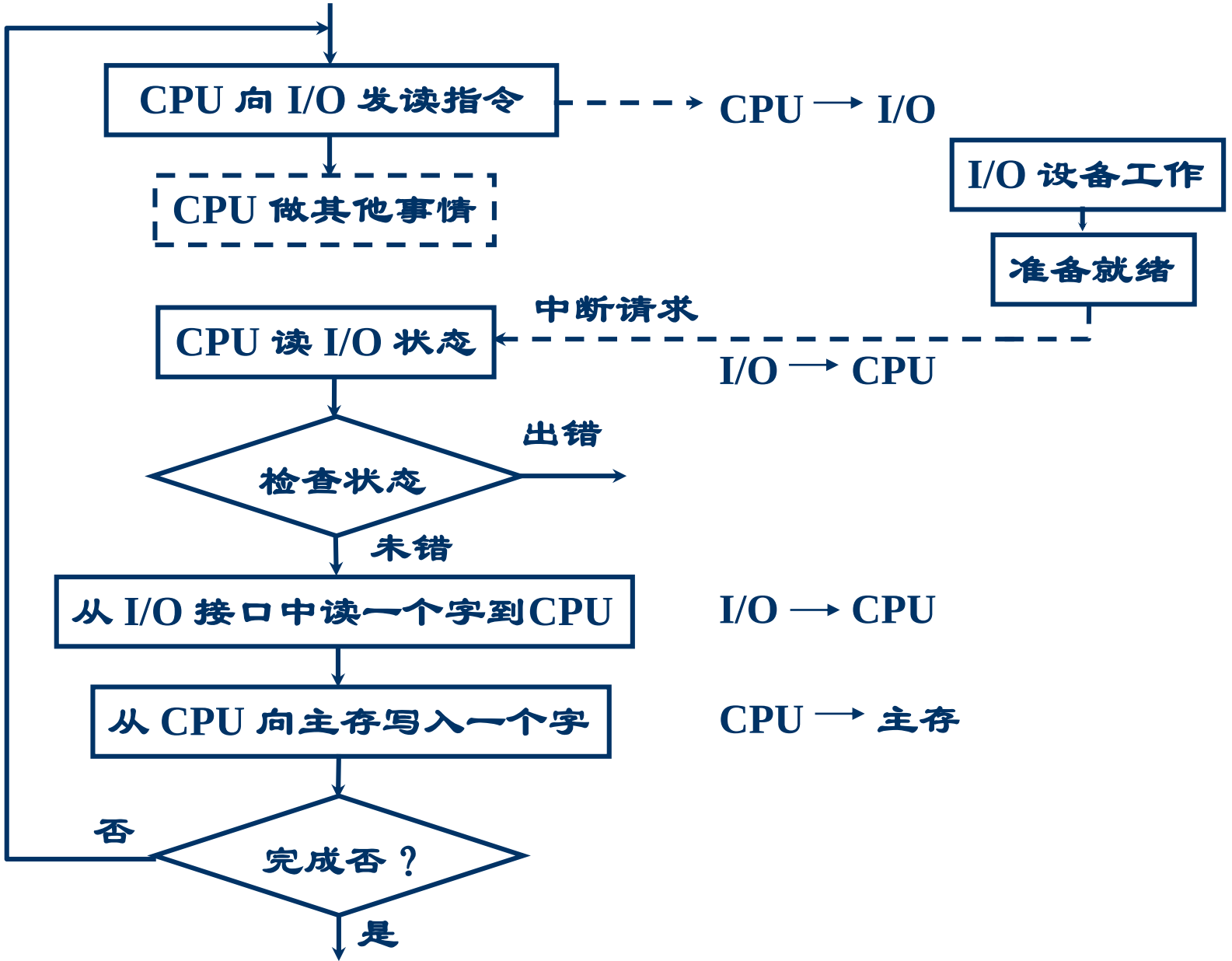
CPU 和 I/O 并行工作



没有踏步等待现象

中断现行程序

程序中断方式流程


CPU 向 I/O 发读指令
CPU → I/O
I/O 设备工作
CPU 做其他事情
准备就绪
中断请求
CPU 读 I/O 状态
I/O → CPU
出错
检查状态
未错
从 I/O 接口中读一个字到CPU
I/O → CPU
从 CPU 向主存写入一个字
CPU → 主存
否
完成否？
是

直接存储器存取（DMA）方式

- 直接存储器存取方式也称为成组传送方式
- 为什么使用DMA方式？
  - 减少大批量数据传输时CPU的开销
    - 可大大减轻CPU负担，CPU只需对DMA控制器进行初期化处理和后处理
  - 解决高速IO设备可能丢失数据问题，满足IO数据交换速度要求
    - 高速IO设备（磁盘等）数据传输速度已经接近于主存储器（DRAM）的工作速度，程序查询和中断方式不能满足要求；
  - 因此，从性能和成本方面综合考虑，必须在IO设备与RAM之间建立直接的数据传送通道，即DMA方式

- DMA方式数据传输特点：
  - 以数据块为单位
  - 主要用于高速的I/O设备，如网卡、磁带、磁盘、模数转换器等设备
  - CPU和外围设备并行工作，且整个数据传送过程不需要CPU的干预
  - I/O和CPU竞争使用存储器
- DMA方式传送数据方法：
  - 采用专用部件（DMA控制器）生成访存地址并控制访存过程，使I/O设备直接和存储器进行成批数据的快速传送
    - DMA控制器将一组数据(块)直接从IO设备送到存储器
    - DMA控制器直接从存储器取出一组数据送到I/O设备

DMA数据传送控制过程

step1、数据传送前，CPU对DMA控制器中控制寄存器初始化

① IO设备准备就绪时，向CPU申请中断服务

② CPU设置DMA控制器的**主存地址寄存器**（主存缓冲区首址）、**设备地址**（如磁盘存储器的物理地址）以及**数据块的长度计数器**

③ 启动DMA设备

step2、 DMA控制器控制设备与存储器直接进行块数据交换

- I/O设备和主存之间在DMA控制器的控制下进行直接数据传输
- 主存地址寄存器随着数据传输的进行而递减/加改变
- 直到主存地址寄存器和计数器到规定值为止

step3、数据传送结束后，CPU进行后处理

- 当计数器的值减到0时，DMA控制器撤销总线请求信号HOLD，整个DMA数据传输过程全部结束。

DMA控制器 （Intel 8237A）的基本结构

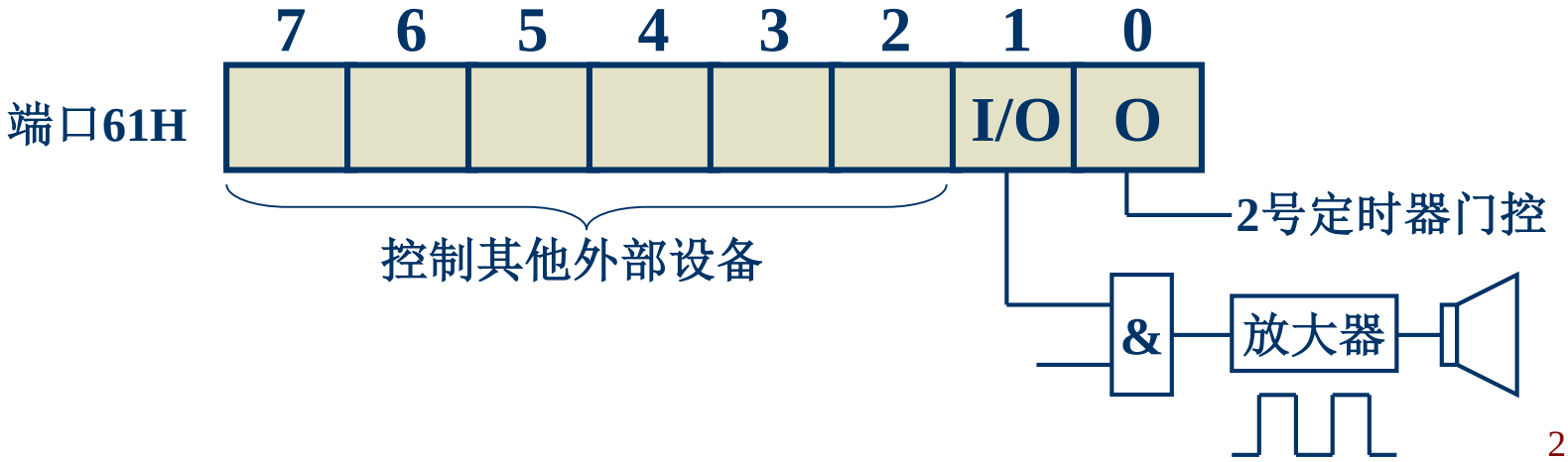
包括4个寄存器：

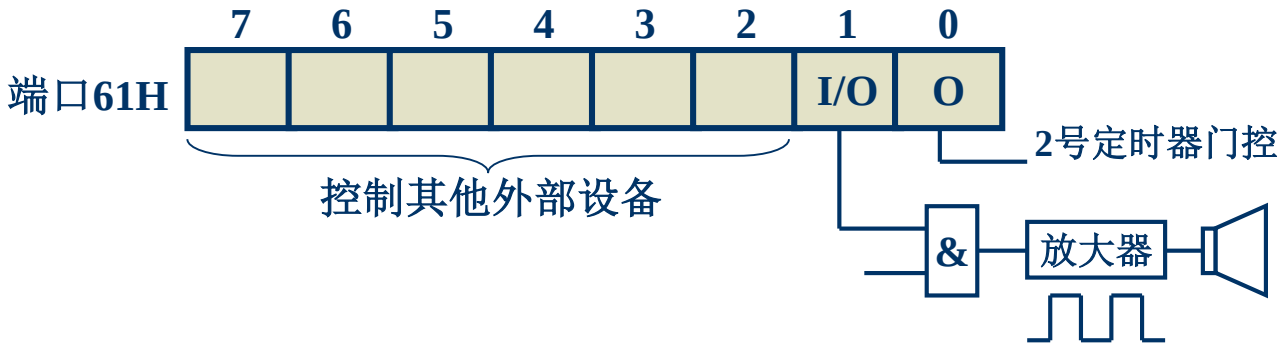
- 控制寄存器
- 状态寄存器
- 地址寄存器
  - 数据块在存储器中启始地址
- 字节计数器
  - 传送数据块的字节数

8.2 I/O程序举例

例8.1 P308： SOUND子程序

- 设备控制寄存器（I/O端口地址为61H）
- 第1位和扬声器的脉冲门相连
- 第1位交替为0和1，就产生了脉冲，根据0和1的延迟控制脉冲频率

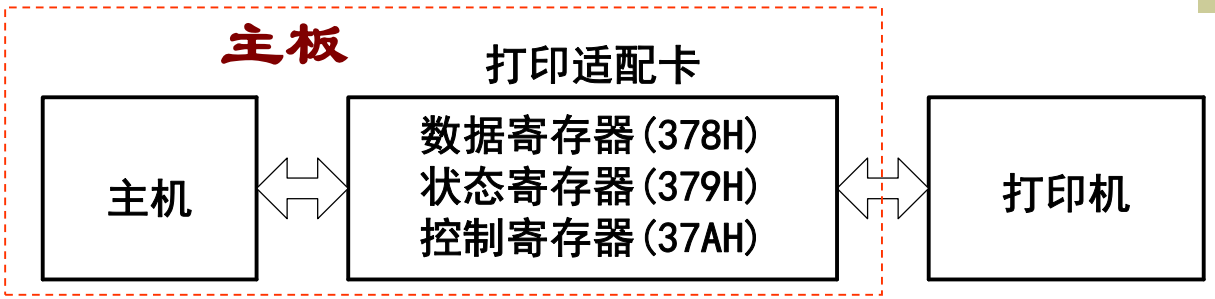





```
; BX: 声音频率（0和1的延迟），  CX：发声时间
;SOUND  PROC    NEAR
        PUSH    AX
        PUSH    DX
        MOV     DX, CX
        IN      AL, 61H
        AND     AL, 11111100B
TRIG:   XOR     AL, 00000010B   ; D1求反，其他位不变
        OUT     61H, AL
        MOV     CX, BX          } 0和1的延迟，输出电平宽度
DELAY:  LOOP    DELAY           }
        DEC     DX
        JNE     TRIG
        POP     DX
        POP     AX
        RET
SOUND   ENDP
```

例8.2　p310：PRT_CHAR程序

- 查询方式打印输出



主机、打印机与接口

状态寄存器(379H)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

D7　忙位（0=忙）
D6　应答（0=可接受）
D5　纸出界（1=纸尽）
D4　联机状态（1=联机）
D3　打印错误（0=出错）
D2、D1、D0　保留未用

控制寄存器(37AH)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

D7、D6、D5保留
D4　控制方式（1=允许中断）
D3　选择位（1=接通打印机）
D2　初始化（0=初始化打印机）
D1　自动换行（1=换行）
D0　数据选通（1=输出数据）

打印机状态寄存器和控制寄存器各位的定义

状态寄存器(379H)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

D7　忙位（0=忙）
D6　应答（0=可接受）
D5　纸出界（1=纸尽）
D4　联机状态（1=联机）
D3　打印错误（0=出错）
D2、D1、D0　保留未用

控制寄存器(37AH)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

D7、D6、D5保留
D4　控制方式（1=允许中断）
D3　选择位（1=接通打印机）
D2　初始化（0=初始化打印机）
D1　自动换行（1=换行）
D0　数据选通（1=输出数据）

打印机状态寄存器和控制寄存器各位的定义

```
data    segment
        mess    db 'Printer is normal', 0dh, 0ah
        count   equ  $-mess
data    ends
cseg    segment
main    proc    far
        assume cs:cseg, ds:data
start:  mov     si, offset mess
        mov     cx, count
next:   mov     dx, 379h
wait:   in      al, dx
        test    al, 80h
        je      wait
        mov     al, [si]
        mov     dx, 378h
        out     dx, al
```

判断打印机是否忙

输出字符到接口的数据寄存器

0dh=00001101b
0ch=00001100b

```
        mov     dx, 37ah
        mov     al, 0dh
        out     dx, al
        mov     al, 0ch
        out     dx, al
        inc     si
        loop    next
        mov     ah, 4ch
        int     21h
main    endp
cseg    ends
        end     start
```

产生数据选通等信号

返回DOS

请回答程序中各指令的寻址方式

8.3 中断传送方式

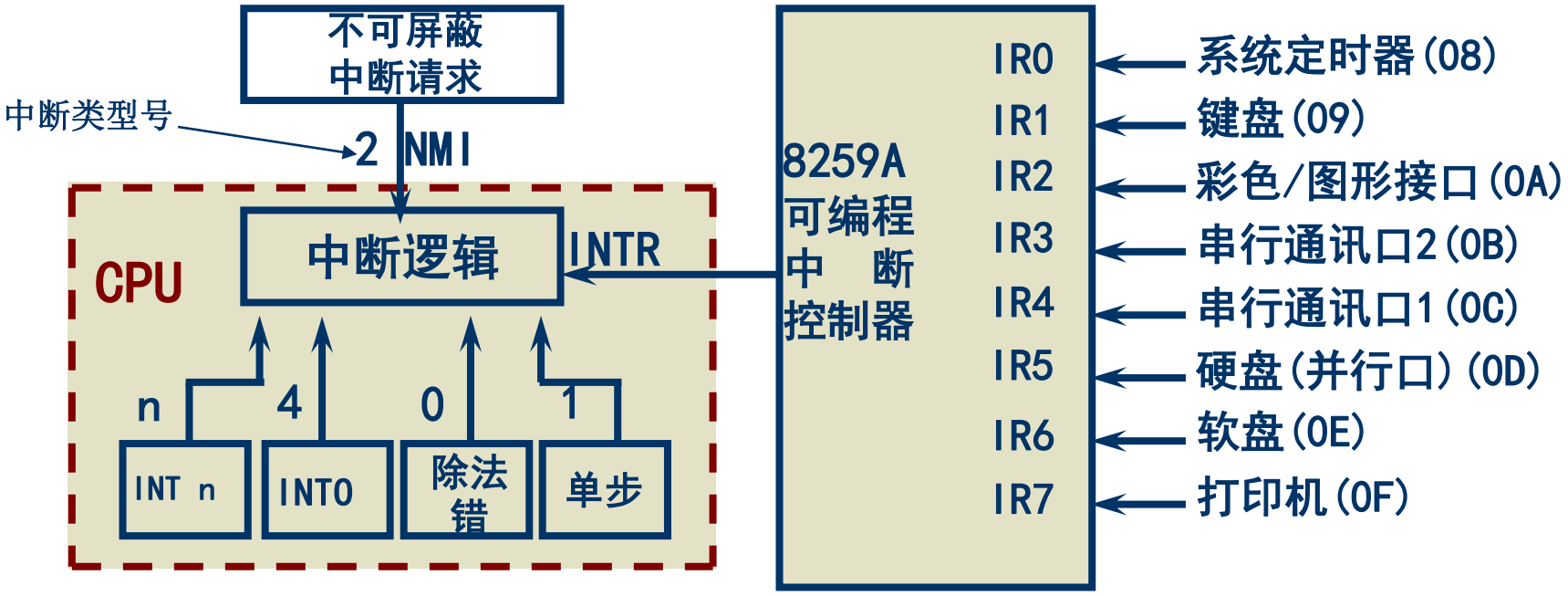
本节主要介绍如下内容：

1. 中断的概念
2. 中断的分类
3. 中断向量表
4. 中断过程
5. 用户自定义中断
6. 中断优先级
7. 中断嵌套

1、中断的概念

- **中断的概念：** 计算机在执行正常程序的过程中，当出现异常事件或事先安排好的事件，迫使CPU暂时中止现行程序的执行，转去执行另一处理程序；当处理完后，CPU再返回到被暂时中止的程序，接着执行被暂时中止的程序。这个过程称为中断
- **中断源：** 引起中断的事件。
  - 如外设输入/输出请求，计算机异常事故或其他内部原因

80X86中断源

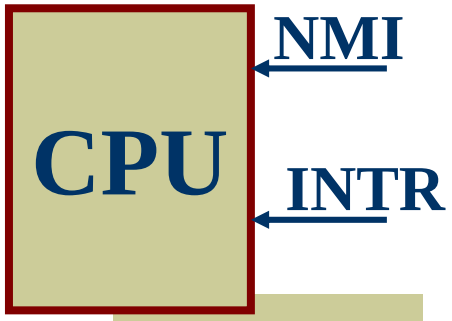


- 不受中断标志位IF屏蔽：
  - 非屏蔽中断(NMI)：电源错、内存和总线奇偶等异常，中断类型号=2
  - 程序中INT指令、运算结果异常等
- 受中断标志位IF屏蔽：
  - 可屏蔽的外部设备中断请求(INTR)

2、中断的分类

- 8086/8088可处理256种中断，对应的中断类型码为0～255
- 按照引起中断的方式，中断可分为：
  ① 硬件中断（或外中断）：外设控制器或协处理器引起的中断
  ② 软件中断或内中断：程序中的中断指令INT或CPU错误结果产生的中断

① 硬件中断



- 硬件中断由外部硬件产生，也称外部中断
- 硬件中断分两类：
  - 非屏蔽中断（NMI：Non Maskable Interrupt）
    - 非屏蔽中断，通过8086/8088的NMI脚引入，它不受中断允许标志IF的屏蔽；中断类型号=2
  - 可屏蔽中断（ Maskable Interrupt）
    - 可屏蔽中断通过CPU的INTR脚引入，只有当标志寄存器的中断屏蔽标志IF为1时，才能引起中断
    - 开中断指令： STI ，设置中断允许位（IF=1）
    - 关中断指令： CLI ，清除中断允许位（IF=0）

- 系统中，通过中断控制器（如8259A）的配合工作，并8259A 的树型连接8086/8088可处理两百多个可屏蔽中断

- 8259A中与中断相关寄存器
  - 中断屏蔽寄存器(IMR) ： 屏蔽单个中断请求
  - 中断命令寄存器

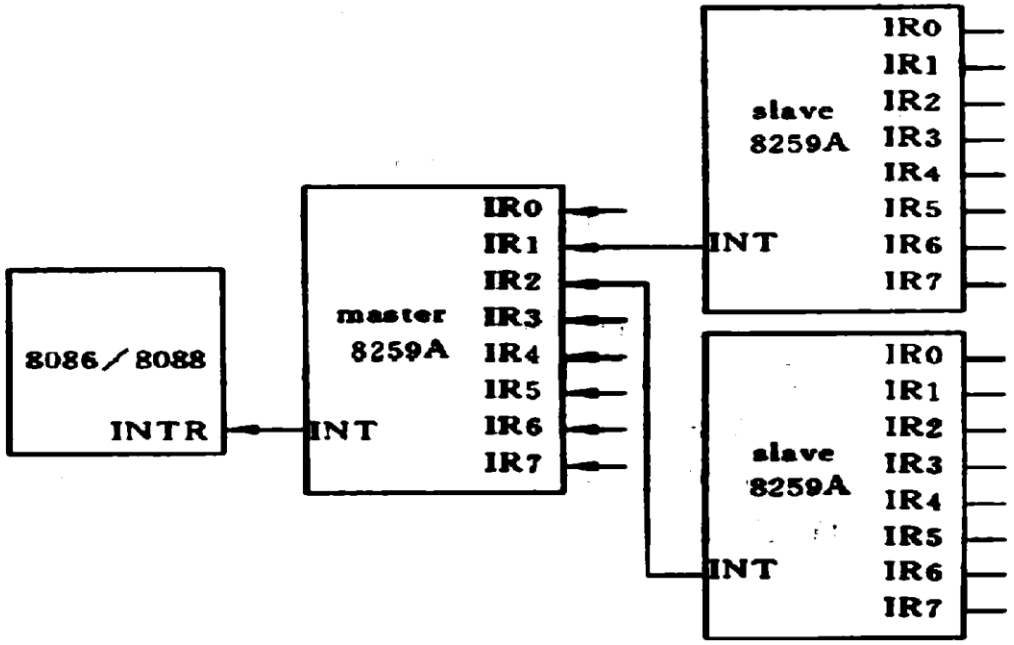
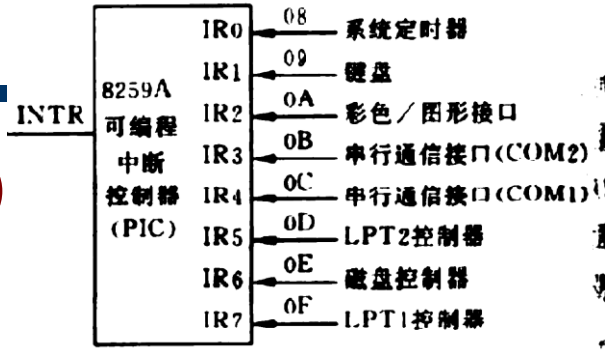


图 8.9 多级 8259A 中断系统



8259A的中断屏蔽寄存器(IMR)

- IMR的I/O端口地址 = 21H
- 8位对应8个外部设备，允许/禁止某设备产生中断

7	6	5	4	3	2	1	0
打印机	软盘	硬盘	COM1	COM2	彩显	键盘	定时器

- =0时，允许中断请求
- =1时，禁止中断请求

例：只允许键盘中断，设置中断屏蔽字

```
MOV   AL,   11111101B
OUT   21H,   AL
```

例：新增设允许键盘中断，设置中断屏蔽字

```
IN   AL,   21H
AND   AL,   11111101B
OUT   21H,   AL
```

CPU可以响应某设备的中断服务请求的条件

- 中断屏蔽寄存器中对应位=0，同时IF=1

屏蔽中断方式

1. 屏蔽所有可屏蔽的外部设备中断请求

- 用标志寄存器中的中断屏蔽标志位IF
  - 开中断指令：STI，设置中断允许位（IF=1）
  - 关中断指令：CLI，清除中断允许位（IF=0）

2. 屏蔽单个可屏蔽的外部设备中断请求

- 用8259A中的中断屏蔽寄存器对应屏蔽位
  - OUT指令

3. 禁止单个中断源的中断请求

- 用设备接口中控制寄存器对应位
  - OUT指令

状态寄存器(379H)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

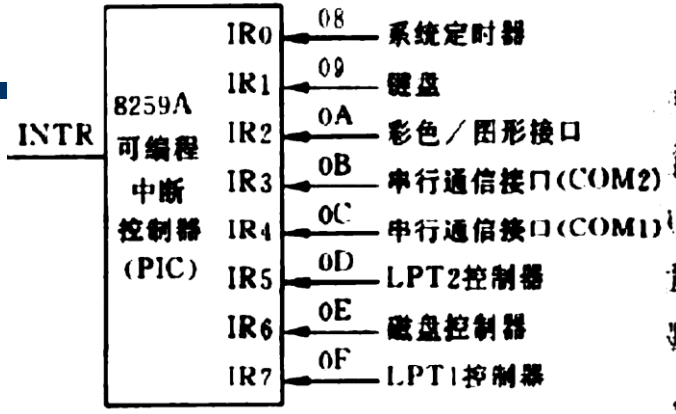
D7 忙位（0=忙）
D6 应答（0=可接受）
D5 纸出界（1=纸尽）
D4 联机状态（1=联机）
D3 打印错误（0=出错）
D2、D1、D0 保留未用

控制寄存器(37AH)各位的定义

D7	D6	D5	D4	D3	D2	D1	D0

D7、D6、D5保留
D4 控制方式（1=允许中断请求）
D3 选择位（1=接通打印机）
D2 初始化（0=初始化打印机）
D1 自动换行（1=换行）
D0 数据选通（1=输出数据）

打印机状态寄存器和控制寄存器各位的定义



中断命令寄存器

- I/O端口地址 = 20H
- 8位含义

7	6	5	4	3	2	1	0
R	SL	EOI	0	0	L2	L1	L0

  - L2-L0: 指定IR0-IR7中哪个中断优先级最低
  - R(rotate), SL(set level)控制IR0-IR7中断优先顺序
  - EOI: 中断结束，当EOI=1时，将当前中断请求清除

- 中断服务程序中，中断处理结束前，应将EOI置1
  - 结束硬件中断指令

```
IN  AL,  20H
OR  AL,  20H
OUT 20H, AL
```

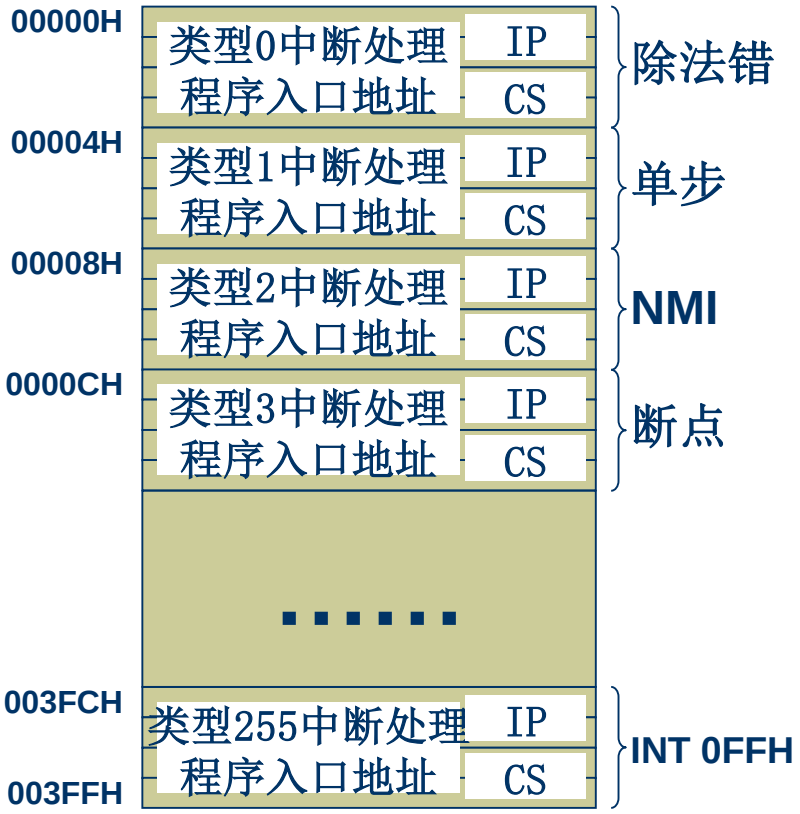
  - 硬件中断服务程序结束前要有这几条指令

② 软件中断

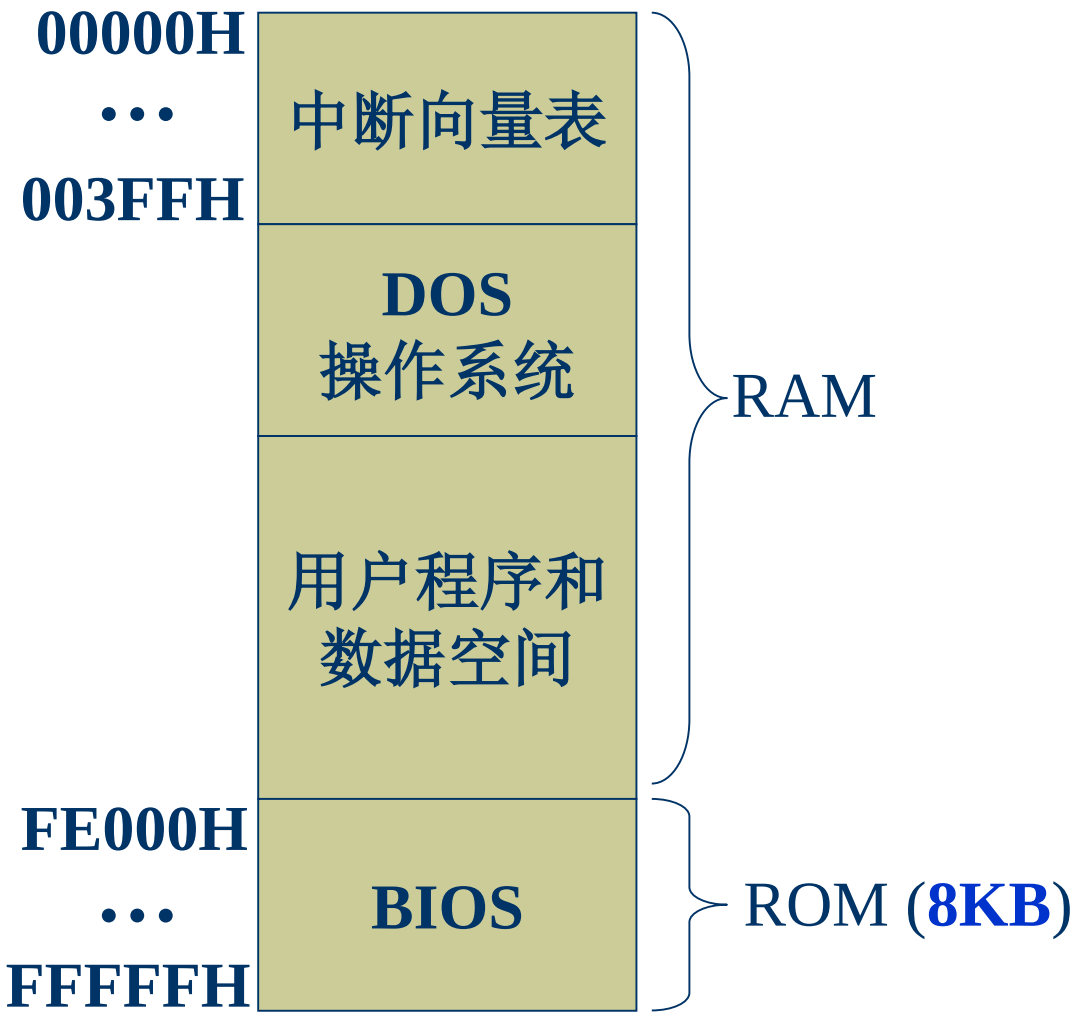
- 软件中断也称内部中断，由3种情况引起
  1. 程序中的中断指令 INT n
     - 操作数n指出中断类型号，0—FFH
     - 如 INT 12H ；存储器容量测试
  2. CPU的某些运行结果
     - 除法错中断：除数为零/商超出表数范围，中断类型号为0的内部中断
     - 溢出中断：运算结果溢出，OF=1，INTO指令将引起类型为4的内部中断
  3. 调试程序（DEBUG）设置的中断
     - 单步中断：标志位TF=1时，中断类型号=1
     - 断点中断：将程序分段，每段设置一个断点（INT 3），中断类型号=3
- 软件中断不受中断屏蔽标志IF影响，不可屏蔽中断

3、中断向量表

- 80X86有256种类型的中断
  - 每种中断有一个中断类型号，类型号 0-FFH
  - 每种类型中断都由相应的中断处理程序处理
- 中断向量表
  - 各类型中断处理程序的入口地址表
  - 保存在存储器中， 1KB (00000H-003FFH)
    - 每类型中断向量占4字节，对应中断处理程序入口CS、IP值
    - 每类中断向量表地址=4×中断类型号n



X86的空间内存分配：


00000H
…
003FFH
中断向量表
DOS
操作系统
用户程序和
数据空间
RAM
FE000H
…
FFFFFH
BIOS
ROM (8KB)

以软中断为例说明中断过程

主程序

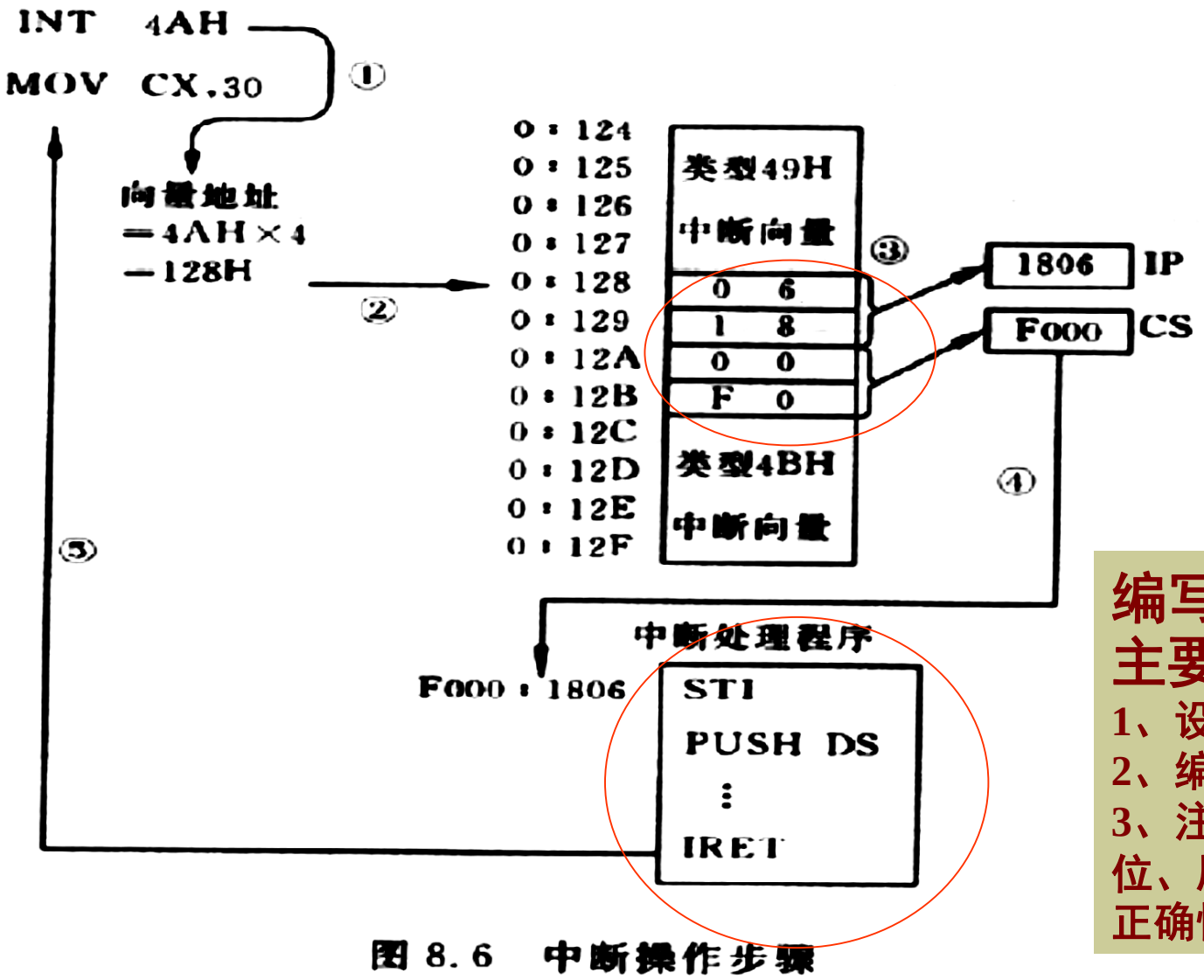


图 8.6 中断操作步骤

编写中断程序时主要工作有：

1、设置中断向量
2、编写中断处理程序
3、注意相关中断允许位、屏蔽位、优先级正确性

地 址	中断类型号	
0~7F	0~1F	BIOS 中断向量
80~FF	20~3F	DOS 中断向量
100~17F	40~5F	扩充 BIOS 中断向量
180~19F	60~67	用户中断向量
1A0~1BF	68~6F	保留
1C0~1DF	70~77	I / O 设备中断向量
1E0~1FF	78~7F	保留
200~3C3	80~FD	BASIC
3C4~3FF	F1~FF	保留

详见附录3，p603

BIOS中断：BIOS中提供的各中断程序
DOS中断：DOS中提供的各中断程序
IO设备中断： 各种IO设备的中断程序

中断类型号的分配

分类	中断类型码	地址（0000H段）	功能
系统内中断（BIOS）	0	0000H~0003H	被零除
	1	0004H~0007H	单步
	2	0008H~000BH	不可屏蔽
	3	000CH~000FH	断点
	4	0010H~0013H	溢出
	5	0014H~0017H	打印屏幕
	6	0018H~001BH	保留
	7	001CH~001FH	保留
分类	中断类型码	地址（0000H段）	功能
系统8级外中断（BIOS）	8	0020H~0023H	日时钟
	9	0024H~0027H	键盘
	A	0028H~002BH	保留
	B	002CH~002FH	异步通信（2）
	C	0030H~0033H	异步通信（1）
	D	0034H~0037H	硬盘
	E	0038H~003BH	软盘
	F	003CH~003FH	打印机

这些功能可以用中断方式调用INT n

分类	中断类型码	地址（0000H段）	功能
设备驱动（BIOS）	10	0040H~0043H	显示
	11	0044H~0047H	设备配制
	12	0048H~004BH	存储容量
	13	004CH~004FH	硬盘I/O
	14	0050H~0053H	通信I/O
	15	0054H~0057H	盒式磁带I/O
	16	0058H~005BH	键盘I/O
	17	005CH~005FH	打印机I/O
	18	0060H~0063H	ROM BASIC
	19	0064H~0067H	系统自举
	1A	0068H~006BH	日时钟I/O
	1B	006CH~006FH	键盘中断地址
	1C	0070H~0073H	定时器报时
	1D	0074H~0077H	显示器参数
	1E	0078H~007BH	软盘参数
	1F	007CH~007FH	图形字符扩展
分类	中断类型码	地址（0000H段）	功能
DOS	20~2F	0080H~00BFH	DOS调用
	30~3F	00C0H~00FFH	为DOS保留

◆ 存取中断向量的DOS功能调用 (21H)

■ 设置中断向量

预置：AH=25H

AL=中断类型号

DS:DX=中断向量（中断程序入口地址）

执行：INT 21H

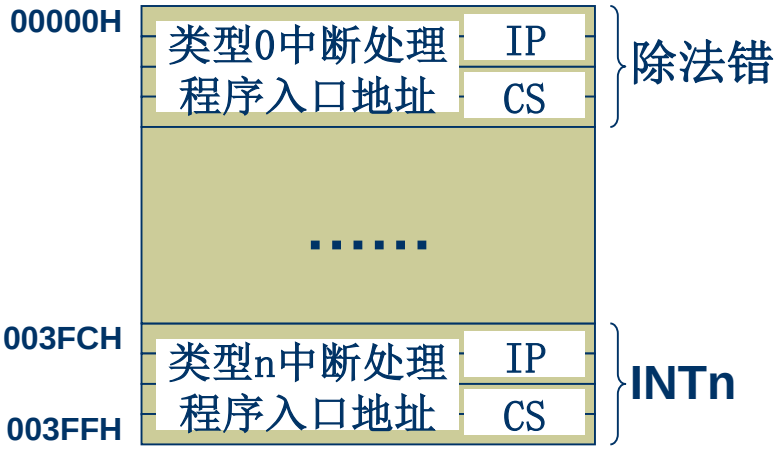
■ 取中断向量

预置：AH=35H

AL=中断类型号

执行：INT 21H

返回：ES:BX=中断向量（中断程序入口地址）



例8.4　使用DOS功能调用存取中断向量

保存原中断向量

```
…
MOV     AL,     N
MOV     AH,     35H
INT     21H
PUSH    ES
PUSH    BX
PUSH    DS
```

设置新中断向量

```
MOV     AX,     SEG   INTHAND
MOV     DS,     AX
MOV     DX,     OFFSET   INTHAND
MOV     AL,     N
MOV     AH,     25H
INT     21H
POP     DS
…
```

恢复原中断向量

```
…
POP     DX
POP     DS
MOV     AL,     N
MOV     AH,     25H
INT     21H
RET
```

拥护自定义中断程序

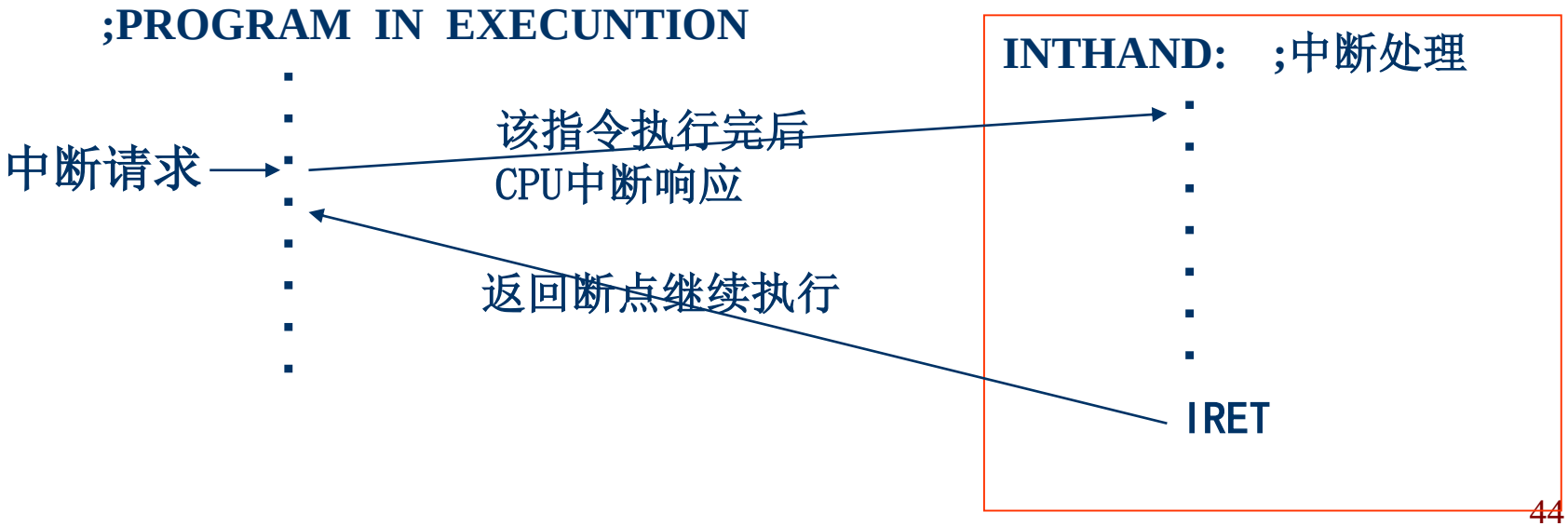
```
INHAND      PROC  FAR
            …
            …
            IRET
```

4、中断过程

① 中断响应

② 中断处理

③ 中断返回



中断响应过程

- 中断响应时，由CPU自动完成如下操作

① 取中断类型号N

② 标志寄存器(FLAGS)内容入栈

③ 保存被中断程序断点
  - 当前代码段寄存器(CS)内容入栈
  - 当前指令指针寄存器(IP)内容入栈

④ 禁止硬件中断和单步中断
  - IF 清 0
  - TF 清 0

⑤ 转中断服务处理程序入口地址
  - 从中断向量表中取4*N单元的字内容 送 IP
  - 从中断向量表中取4*N+2单元的字内容 送 CS

注意：如果在中断处理程序中允许处理可屏蔽中断，要重新设置IF为1

中断处理过程
（中断处理程序）

- 中断功能处理与子程序功能处理类似，但注意几点
  - 保存现场：中断发生是不可预知的，除CS、IP、SS、SP外，一般凡用到的寄存器都应保存入堆栈
  - 如果中断处理程序中允许外部中断，要重新设置IF为1
- 一般中断处理程序设计格式
  1. 保存现场
  2. 开中断(STI)　（根据需要确定）
  3. 中断处理程序主体部分
  4. 中断结束（EOI）（硬件中断时）
  5. 关中断(CLI)
  6. 恢复现场
  7. 中断返回(IRET)

中断返回过程

- 中断返回时（IRET），由CPU自动完成如下操作

① 恢复被中断程序断点程序指针

- 被中断程序指令指针寄存器(IP)内容恢复
- 被中断程序代码段寄存器(CS)内容恢复

② 标志寄存器(FLAGS)内容从栈恢复

中断与子程序调用

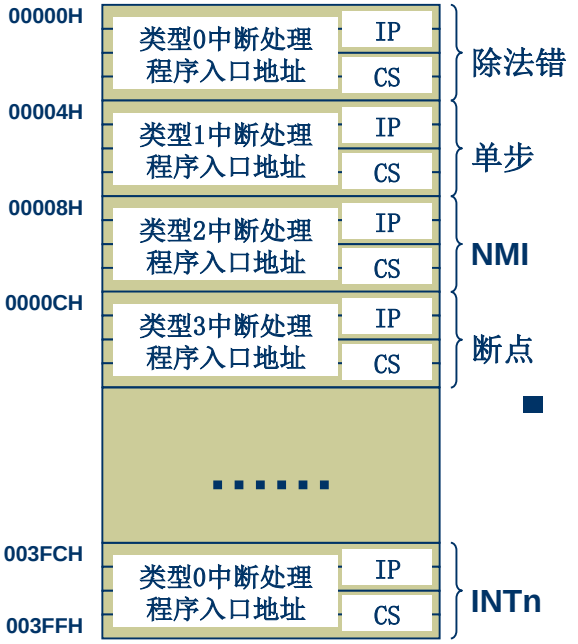
- 中断与子程序调用处理过程相似
- 差别主要在于进入和返回时的处理不同
  - 进入中断服务处理程序时
    - 子程序调用：只把CS和IP压入堆栈
    - 中断：除把CS和IP压入堆栈外，还把标志寄存器的内容压入堆栈，并且关掉了中断和单步运行方式
  - 返回时
    - 子程序返回：只把断点地址从堆栈弹出送CS和IP
    - 中断返回：除恢复断点地址CS和IP外，还要恢复标志寄存器的内容
- 时机不同：
  - 中断：一般随机发生；软中断在程序中预先安排
  - 子程序调用：程序中预先安排调用

5、用户自定义中断

- 用户可以用保留的中断类型，扩充自定义中断功能
- 新增加中断时
  - 编写中断服务处理程序
  - 主程序中：
    - 设置中断向量表中相应的中断向量
    - 正确设置中断允许、屏蔽位和优先级

地 址	中断类型号	
0～7F	0～1F	BIOS 中断向量
80～FF	20～3F	DOS 中断向量
100～17F	40～5F	扩充 BIOS 中断向量
180～19F	60～67	用户中断向量
1A0～1BF	68～6F	保留

地 址	中断类型号	
1C0～1DF	70～77	I / O 设备中断向量
1E0～1FF	78～7F	保留
200～3C3	80～FD	BASIC
3C4～3FF	F1～FF	保留



◆ 例如：用户自定义中断类型n

```
; 设置中断向量
   MOV   AX,  0
   MOV   ES,  AX          ; 设置中断向量基地址
   MOV   BX,  N*4         ; 设置中断向量n偏移地址
   MOV   AX,  OFFSET  INTHAND
   MOV   ES:WORD PTR[BX],  AX    ; 设置中断处理程序入口地址偏移量
   MOV   AX,  SEG  INTHAND
   MOV   ES:WORD PTR[BX+2],  AX  ; 设置中断处理程序入口段基地址
   ..... ; 等待中断请求，  或者使用软中断指令INT n进入中断服务子程序
```

也可以用DOS调用设置新中断向量

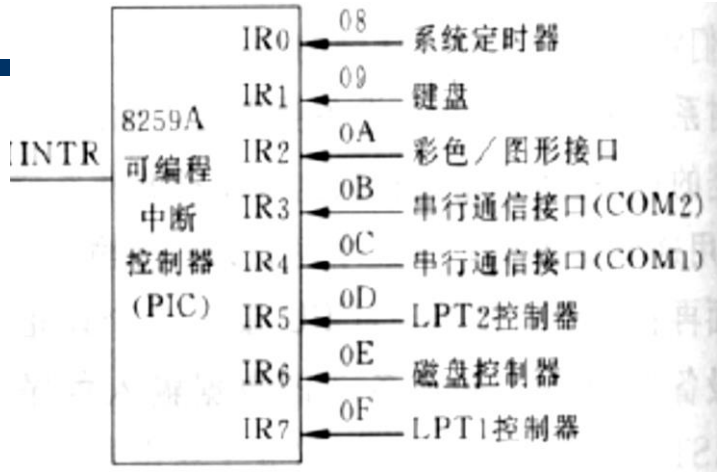
```
; 中断处理程序
INTHAND   PROC   FAR
   .....      ; 中断服务功能处理
   IRET
```

6、中断优先级

- 当多个中断源同时向CPU请求中断时，CPU应如何处理？
  - 制定优先级别，先为优先级别高的中断服务
- **中断优先级(Priority)：** 当多个中断源同时向CPU请求中断时，CPU确定优先处理的次序
- 80X86中规定的中断优先级次序

优先级高 ↓ 低

  - 软件中断（除法错，INTO，INT）
  - 非屏蔽中断（NMI）
  - 可屏蔽中断（INTR）
  - 单步中断



- **可屏蔽中断优先级分8级**
  - 由8259A中断控制方式确定
  - 正常默认次序由高到低为：

IR0, IR1, IR2, IR3, IR4, IR5, IR6, IR7

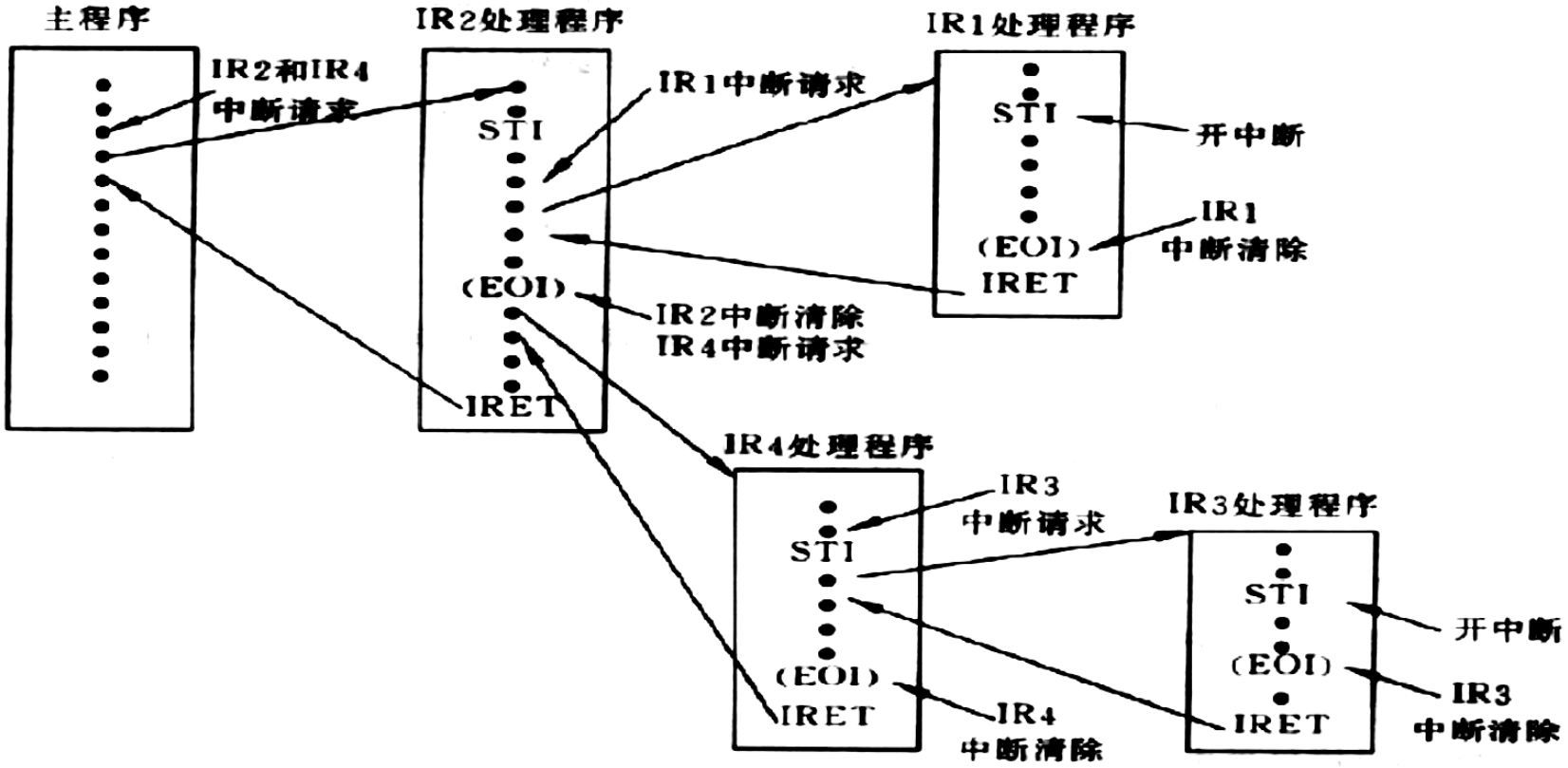
- 8259A的中断命令寄存器的6、7位可控制优先级次序

7	6	5	4	3	2	1	0
R	SL	EOI	0	0	L2	L1	L0

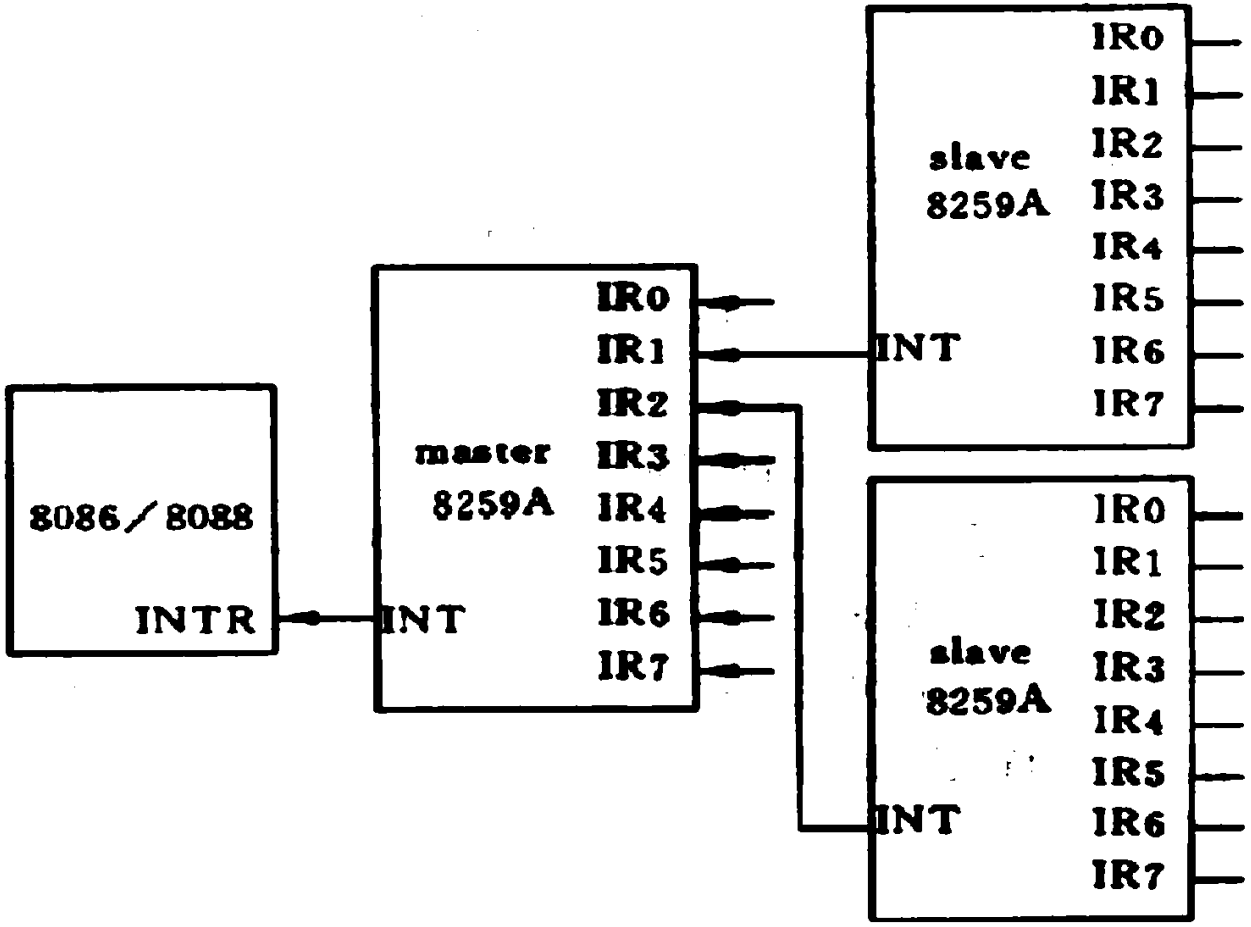
  - R, SL=00时， 正常优先级方式
  - R, SL=01时， 清除由L2-L0指定的中断请求
  - R, SL=10时， 各中断优先级依次左循环一个位置
  - R, SL=11时， 各中断优先级依次左循环，直到由L2-L0指定的中断源的优先级最低
- 根据需要给端口20H送命令， 改变优先级

7、中断嵌套

- **中断嵌套：** 正在运行的中断处理程序，又被其他中断源的中断请求中断



正常优先级方式下的典型中断序列



多级 8259A 中断系统

最高优先级

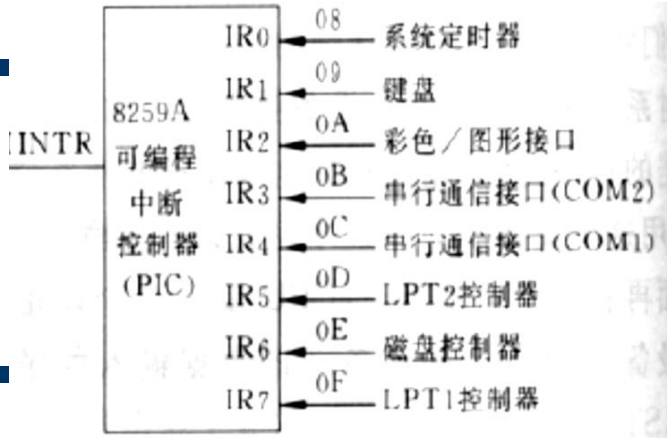
最低优先级

主8259A:IR0，

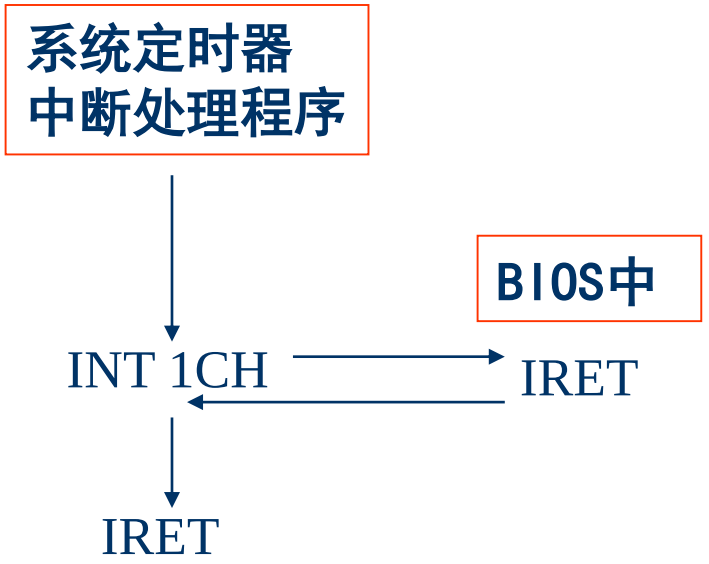
从8259A: IR0, IR1, IR2, IR3, IR4, IR5, IR6, IR7， IR0, IR1, IR2, IR3, IR4, IR5, IR6, IR7，

主8259A: IR3, IR4, IR5， IR5， IR7

中断程序举例



- 例8.5： p323
  - 要求每10秒响铃一次，并显示“The bell is ring!”
  - 要点：如何设计中断处理程序；如何进入中断处理程序
  - 可用资源：系统定时器（中断类型8，每秒中断18.2次）
    - 系统定时器的中断处理程序中，有一条指令INT 1CH，但嵌套调用后BIOS中只有IRET指令。用户可实现完成某些周期性工作，但影响系统时钟

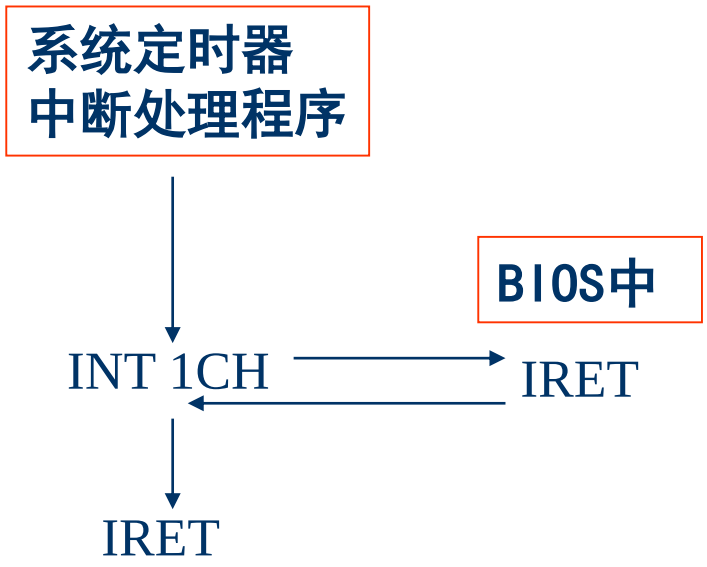


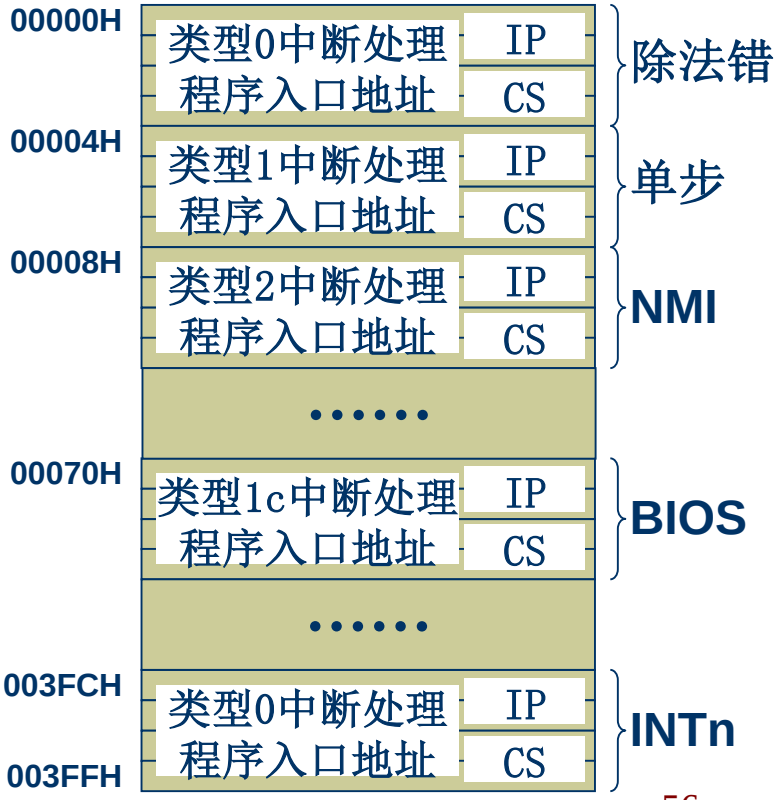
- 中断类型1CH作为用户使用的中断类型，可能已被其他功能的程序使用，所以在编写新的临时中断程序时，应做如下工作：

  1、在主程序的初始化部分，先保存当前1CH的中断向量，再设置新的中断向量

  2、在主程序结束部分恢复保存的1CH中断向量

1ch × 4 = 70h

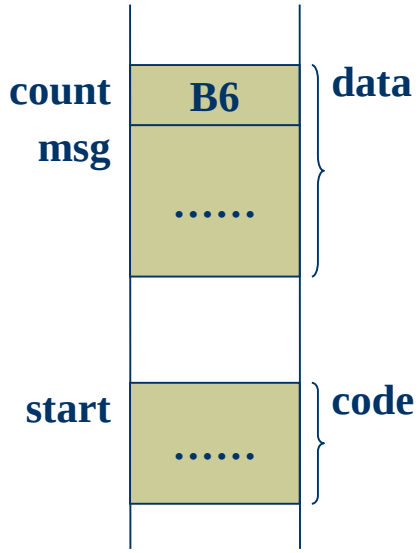




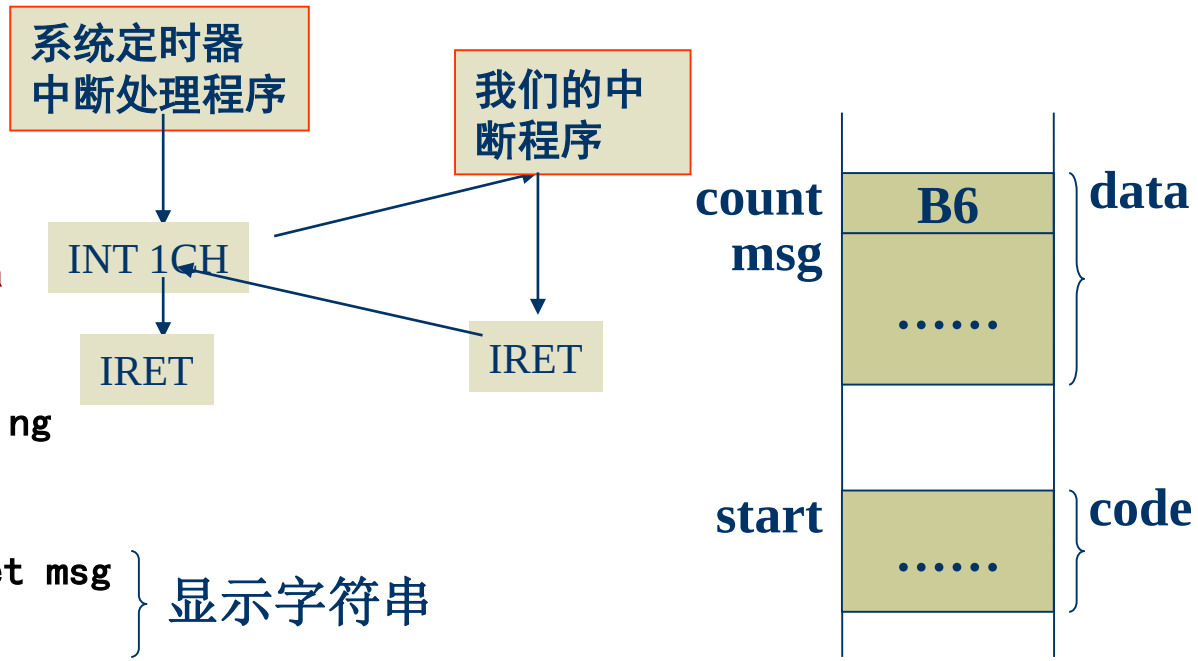
- **设置中断向量**
  - 预置：AH=25H
    AL=中断类型号
    DS:DX=中断向量(中断程序入口地址)
  - 执行：INT 21H
- **取中断向量**
  - 预置：AH=35H
    AL=中断类型号
  - 执行：INT 21H
  - 返回：ES:BX=中断向量(中断程序入口地址)

7	6	5	4	3	2	1	0
打印机	软盘	硬盘	COM1	COM2	彩显	键盘	定时器

```
;*****************************
;eg8-5.asm
;purpose:ring ang display a message every 10 seconds.
;*****************************
          .model samll
;-----------------------------------------
          .stack
;-----------------------------------------
          .code
; main program
main      proc      far
start:    move      ax, @data
          mov       ds, ax

;save old interrupt vector
          mov       al, 1ch
          mov       ah, 35h
          int       21h
          push      es
          push      bx

;set new interrupt vector
          push      ds
          mov       dx, offset ring
          mov       ax, seg ring
          mov       ds, ax
          mov       al, 1ch
          mov       ah, 25h
          int       21h
          pop       ds
          in        al, 21h
          and       al, 11111110b
          out       21h, al
          sti

          mov       di, 20000
deplay:   mov       si, 30000
deplay1:  dec       si
          jnz       deplay1
          dec       di
          jnz       deplay

;restore old interrupt vector
          pop       dx
          pop       ds
          mov       al, 1ch
          mov       ah, 25h
          int       21h

          mov       ax, 4c00h
          int       21h

main      endp
```



```
;-----------------------------------------
;Procedure ring
;Purpose:ring every 10 seconds when substitued for interrupt 1ch
ring    proc    near
        push    ds
        push    ax
        push    cx
        push    dx

        mov     ax, @data
        mov     ds, ax
        sti

;Siren if it is time for ring
        dec     count
        jnz     exit

        mov     dx, offset msg
        mov     ah, 09h
        int     21h

        mov     dx, 100
        in      al, 61h
        and     al, 0fch
sound:  xor     al, 02
        out     61h, al

        mov     cx, 1400h
wait1:  loop    wait1
        dec     dx
        jne     sound
        mov     count, 182
```



显示字符串

响铃

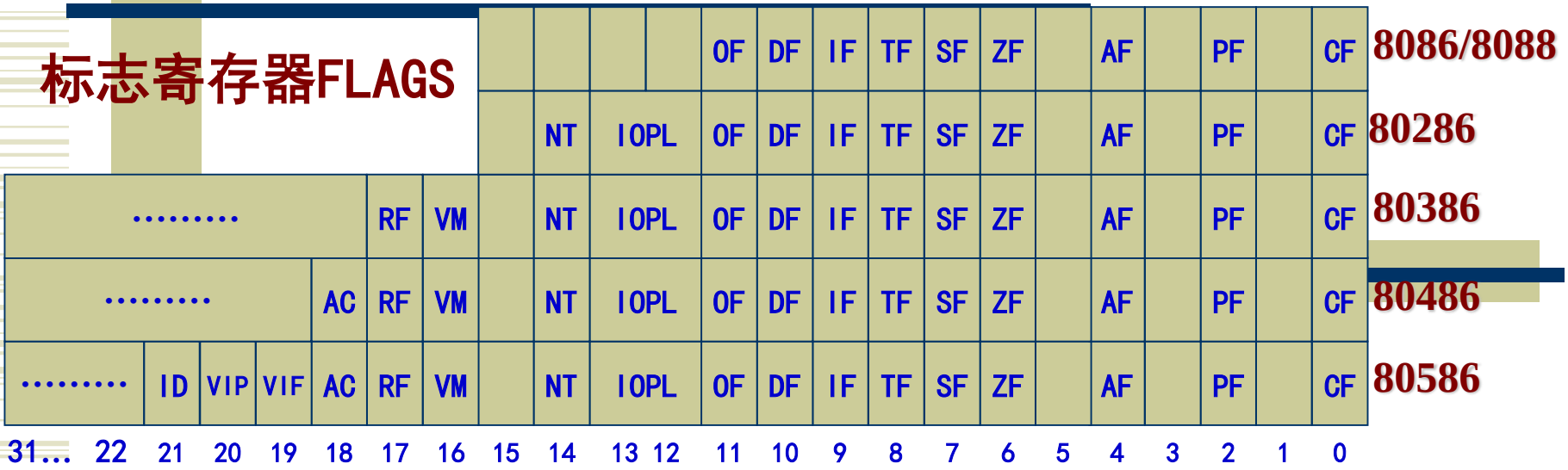
```
exit:   cli
        pop     dx
        pop     cx
        pop     ax
        pop     ds
        iret
ring    endp
;-----------------------------
        end     start
```

8.4 80386输入输出

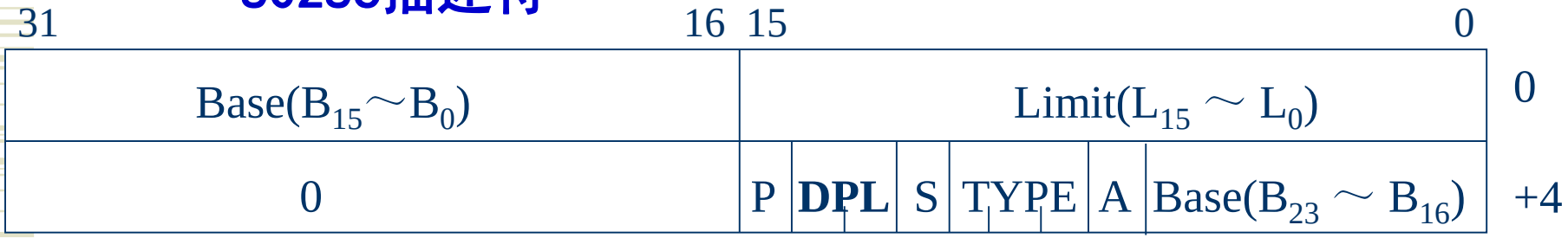
- 支持65536个I/O端口
- 同样IN、OUT指令只能使用累加器(Al,AX,EAX)与外设通信
- 在保护模式下，I/O指令带有特权级，满足特权级才可执行I/O操作
  - 对IO操作的特权级由EFLAGS中的IOPL指定
  - 过程的CPL高于或等于IOPL（CPL≤IOPL）才能执行I/O指令
  - 低特权级过程执行I/O指令，会产生保护故障（中断13）
  - 只有0特权级过程才能修改IOPL
- V86方式下，IOPL仅保护中断标志IF，I/O端口通过TSS中的I/O允许位保护

.....	RF	VM		NT	IOPL	OF	DF	IF	TF	SF	ZF		AF		PF		CF

标志寄存器FLAGS

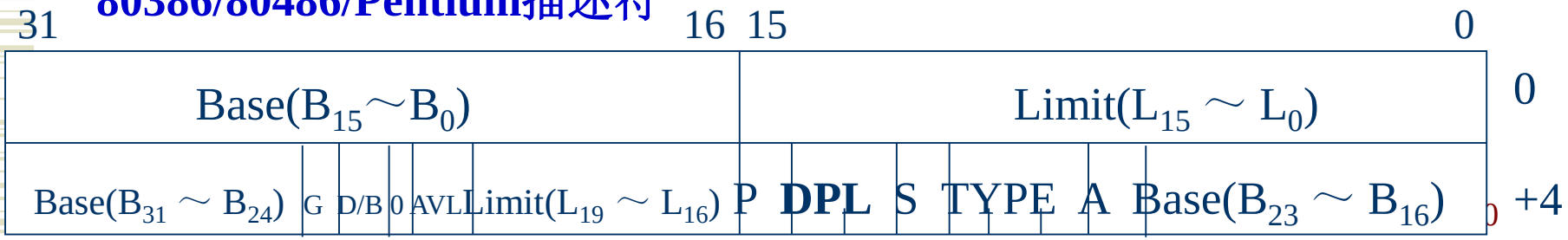


80286描述符



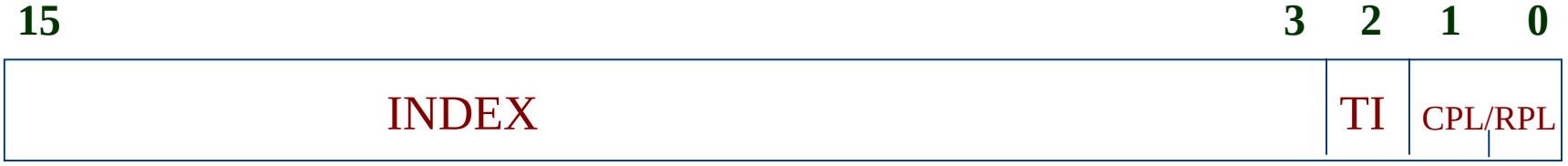
DPL：特权级（0-3）

80386/80486/Pentium描述符



特权级

1. RPL（Requested PL）请求特权级。保存在段选择符的b_1b_0位。表示本次访问所要求的特权级。
2. DPL（Discriptor PL），描述符特权级。它表示该段或任务门的特权级，分别被保存在段描述符和门描述符的DPL域中，段描述符的DPL具体在访问权字节中的b_6b_5位。
3. CPL（Current PL），当前特权级。CPL是当前运行程序或任务的特权级，分别被保存在段寄存器CS即段选择符的b_1b_0位和当前任务寄存器的b_1b_0位。
4. IOPL（IO PL），对IO操作的特权级。保存在EFLAGS中



- 80386还支持字符串I/O指令
  - INSB/OUTSB ；输入输出字符串
  - INSW/OUTSW ；输入输出字串
  - INSD/OUTSD ；输入输出双字串
- INS指令： DX指定端口地址，结果存于ES:EDI存储单元，按DF指定方向修改EDI
- OUTS指令： DX指定端口地址，源数据存于ES:ESI存储单元，按DF指定方向修改ESI
- 可加重复前缀REP，ECX中为重复计数值

I/O允许位图

- 要完成I/O操作，必须满足两个条件之一：
  - I/O特权级（CPL≤IOPL）
  - 任务的I/O允许位图中的端口访问许可位
    - I/O允许位图保存在任务的状态段TSS
      - 对应端口的位=0，可以访问该端口
      - 对应端口的位=1，不可以访问该端口

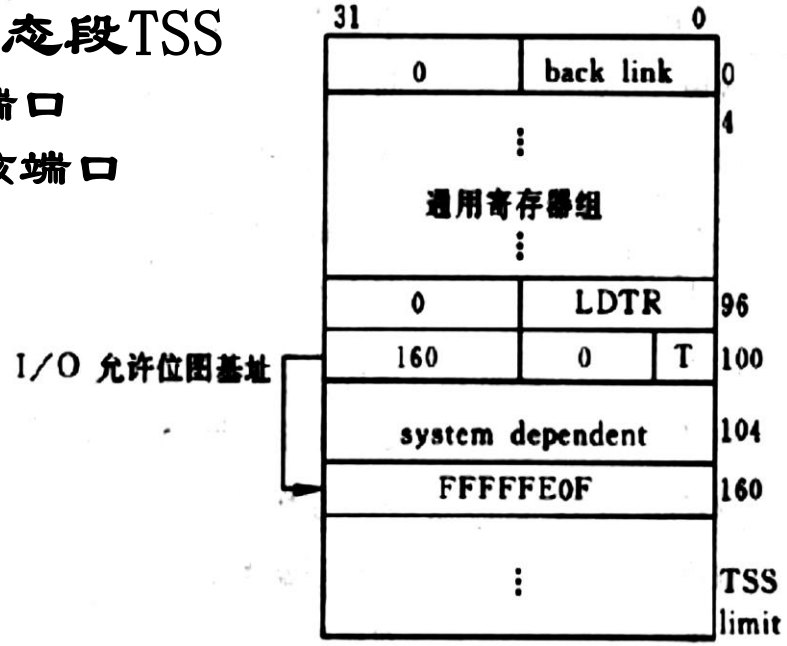


图8.10 TSS中I/O允许位图实例

8.5 80386的中断处理

- 支持3种模式下的中断处理
  - 实模式下与8086中断操作相同
  - 保护模式下，中断寻址根据中断描述符表IDT，每类中断对应IDT中8字节的门描述符
  - 虚拟8086模式下，能模拟虚拟8086的操作方式，但对应保护模式，不对应实模式

8.5.1 80386的中断与异常

中断号	类 型	描 述
0	故障	除错误(仅 DIV、IDIV)
1	故障或陷阱	调试程序中断
2	中断	非屏蔽中断(NMI)
3	陷阱	断点中断
4	陷阱	溢出中断(INTO)
5	故障	超出数组边界(BOUND)
6	故障	无效操作码
7	故障	协处理器无效
8	失败	双故障
9	失败	协处理器段溢出运行(在 80486 上保留)
10	故障	无效 TSS
11	故障	段不存在
12	故障	堆栈段溢出
13	故障	通用保护故障
14	故障	负故障
15	保留	
16	故障	协处理器错误
17	故障	调准检查(仅用于 486,386 保留)
18～30	保留	
31～255	中断或陷阱	系统相关

8.5.2 实地址下的中断处理

- 与8086相同
- INT指令或其他中断请求发生时，自动判断段的寻址方式： 16位、32位
- 中断返回时：
  - IRET指令： 恢复IP、FLAGS
  - IRETD指令： 恢复EIP、EFLAGS

8.5.3 保护方式下的中断

- 中断和异常根据处理方法分为：
  - 中断门(interrupt gate)、陷阱门(trap gate)、任务门(task gate)
- 门描述符
  - 8个字节
  - 寻址相应中断处理程序入口
  - 访问权保护检查

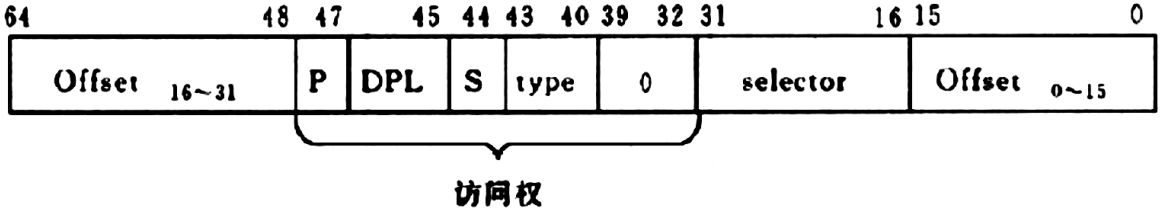
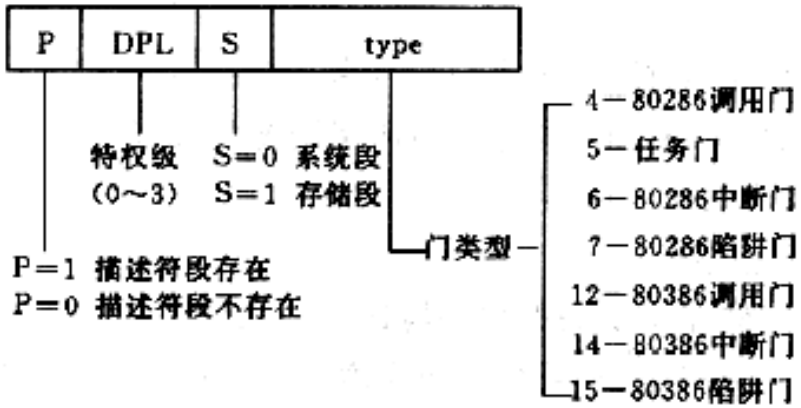
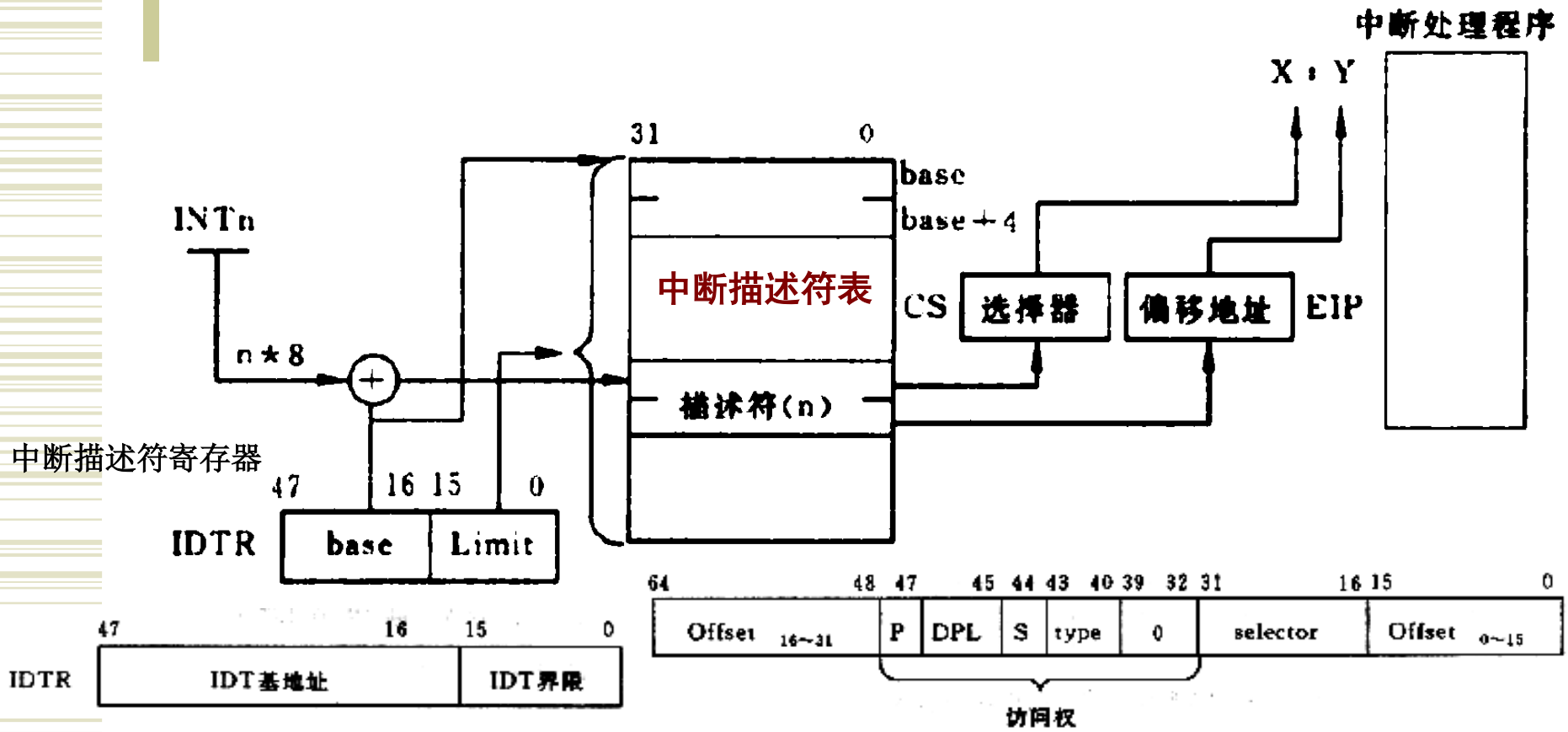


图 8.12 中断门和陷阱门描述符



- 对应任务门时，任务切换
- Selector：16位代码段选择器，直接解释为段基地址
- Offset：32位段内偏移地址

保护模式下中断过程示意



- 中断描述符表相当于实模式下的中断向量表
- 根据中断门描述符获得中断处理程序入口

作　业

8.1　　8.6

8.8　　8.11